

**In the Claims:**

Please cancel claims 4-13, 17-26 and 30-39, without prejudice and without waiver of subject matter set forth therein.

Please amend claims 1, 14 and 27 as follows:

1. A method of translating compiled programming code from a first compiled code state to a second compiled code state, the programming code in the first compiled code state comprising a plurality of basic blocks, each basic block comprising a set of instructions, at least one basic block ending in a dynamic branch, the dynamic branch being a transfer to one of a set of destinations based on a calculation of a destination address, the method comprising the steps of:

identifying the plurality of basic blocks in the first compiled code state of the programming code;

identifying links between the identified basic blocks;

constructing a control flow graph / representation (CFG) of the programming code based on the identified basic blocks and identified links, the CFG being in a preliminary form;

identifying at least one basic block ending in a dynamic branch;

exploring, based on the CFG, and without information from a source external to the first compiled code state of the programming code, all identified basic blocks that lead to the dynamic branch as far back as is necessary to fully determine a set of destination addresses for the dynamic branch, the set of destination addresses defining the set of destinations from the dynamic branch;

examining the set of destinations to identify a branch table;

updating the CFG to reflect the set of destinations and the identified branch table; and

translating the programming code from the first compiled code state to the second compiled code state based at least in part on the updated CFG.

14. A translator operating on a processor for translating compiled programming code from a first compiled code state to a second compiled code state, the programming code in the first compiled code state comprising a plurality of basic blocks, each basic block

comprising a set of instructions, at least one basic block ending in a dynamic branch, the dynamic branch being a transfer to one of a set of destinations based on a calculation of a destination address, the translator:

- identifying the plurality of basic blocks in the first compiled code state of the programming code;

- identifying links between the identified basic blocks;

- constructing a control flow graph / representation (CFG) of the programming code based on the identified basic blocks and identified links, the CFG being in a preliminary form;

- identifying at least one basic block ending in a dynamic branch;

- exploring, based on the CFG, and without information from a source external to the first compiled code state of the programming code, all identified basic blocks that lead to the dynamic branch as far back as is necessary to fully determine, a set of destination addresses for the dynamic branch, the set of destination addresses defining the set of destinations from the dynamic branch;

- examining the set of destinations to identify a branch table;

- updating the CFG to reflect the set of destinations and the identified branch table; and

- translating the programming code from the first compiled code state to the second compiled code state based at least in part on the updated CFG.

27. In connection with translating compiled programming code from a first compiled code state to a second compiled code state, the programming code in the first compiled code state comprising a plurality of basic blocks, each basic block comprising a set of instructions, at least one basic block ending in a dynamic branch, the dynamic branch being a transfer to one of a set of destinations based on a calculation of a destination address, a computer- readable medium having computer-executable instructions for performing steps comprising:

- identifying the plurality of basic blocks in the first compiled code state of the programming code;

- identifying links between the identified basic blocks;